LLNL-TR-412434

Progress report for FACETS (Framework Application for Core-Edge Transport Simulations)

T. G. W. Epperly

April 22, 2009

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Progress report for

FACETS (Framework Application for Core-Edge Transport Simulations)

T. G. W. Epperly, CS SAP Principal Investigator

Lawrence Livermore National Laboratory

April 20, 2008

(DOE Contract DE-AC52-07NA27344)

1. Introduction

This role of this computer science SAP is to facilitate FACETS design and development by contributing CCA component technology and new application-specific technology. From a software perspective, the FACETS project is a very complex project. It is a combination of legacy software written in Fortran, Python, and C++ by various coding groups along with new software modules being written from scratch. The FACETS team is spread among 11 organizations and is geographically distributed from coast to coast. The fusion physics modules to be incorporated vary in terms of the model dimensions, typical time scale, and type of interactions with other components.

Because FACETS is a complex project, it requires a component-based framework to facilitate the definition and composition of scientific applications from a suite of available fusion physics components. Component architectures have proven themselves in the business world and more recently in the scientific computing world. The CS SAP contributes fundamental tools like Babel to the FACETS framework and helps develop application-specific interfaces appropriate for the fusion physics modules.

2. Project Funding, Initial Scope, and Changes

The Center for Applied Scientific Computing at Lawrence Livermore National Laboratory receives $200,000 annually in funding to support 0.4 FTEs. The initial milestones from the initial proposal for FY08 included:

- Define basic FACETS software architecture
- Define component configuration, serialization, and deserialization interfaces to allow startup, initialization, and centralized checkpointing.
- Prototype simplified CCA compliant framework to allow initial coupling of core and edge models
- Prototype framework parallel communication management

The milestones from the initial for FY09 included:

- Define core & edge component interfaces with ports to allow connections between them
- Link component interfaces to core and edge codes
- Demonstrate core and edge components running in framework

- Link models using parallel coupling technology from other project participants
- Assist in linking physics components to TOPS solver components
- Assist in porting framework to required high-performance platforms.

For the most part, the project has proceeded according to our initial plans. We succeeded in defining the initial architecture and component interfaces. We implemented the initial framework and succeeded in getting a coupled core edge simulation running inside FACETS. Other team members have succeeded in linking Uedge with a TOPS solver, Petsc.

The framework is designed to manage parallel physics components, and it can run parallel models. We have not yet needed to incorporate Larson's model coupling toolkit because the interactions between models have not yet required that level of sophistication.

I have spent more time than expected working on incorporating Uedge into the framework and less time working on framework implementation and design. Tech-X Corporation has lead the framework design and implementation, and I have reviewed it and suggested modifications to the design. We have not focused on making FACETS into a CCA compliant rather we have focused on leveraging the parts of the CCA that bring the best value to FACETS.

3. Packaging Uedge as a FACETS component

In the first half of the FACETS project, much of my work has focused on connecting Uedge to the FACETS framework. Connecting Uedge to the FACETS framework turned out to be more challenging than initially estimated due to its commitment to using a scripting language front end. Because of the challenge, we developed a two-phase approach. The first phase was designed to get Uedge up and running quickly inside the FACETS framework, and the second phase was designed to produce a faster interface that did not depend on a scripting language engine.

Uedge defines a scripting language interface using `.v` files. The `.v` files are the input language to a scripting language called Basis and a Python-wrapping tool called Forthon. In the first phase, the approach was to generate a Python-wrapped Uedge using Forthon and then to wrap that version of Uedge with Babel to provide the C++ interface required by FACETS. This work required writing the FACETS interface using Babel's Scientific Interface Definition Language (SIDL), making a build system to generate and build the Babel-generated glue code, and writing the code to link the Babel-generated code to Uedge. This approach allowed the FACETS to produce early results from a couple core-edge fusion reactor simulation.

The second phase began late in the first year and has carried on through the present. In this approach, we created an extension to Forthon that generates Babel interfaces for Uedge directly from the `.v` files. These Babel-based interfaces connect the FACETS framework in C++ to the Uedge physics module in Fortran. The purpose of this approach was to remove Uedge's dependence on a scripting language engine and to facilitate having a statically linked FACETS executable (i.e., no dependence on shared or dynamically loadable libraries). The trend in LCF platforms is to have simple compute nodes with stripped down operating systems, so having a static executable is a requirement for some LCF machines.

Generating a Python-free version of Uedge proved to be very challenging due to Uedge's reliance on Python and 3rd party Python libraries. For example, the checkpoint/restore features in Uedge had to be rewritten in Fortran. A new build system had to be written to compile Uedge without using Forthon. Our approach allows the Uedge team to support FACETS without introducing any new development files because we reuse their existing interface specification for Basis & Forthon.

The Uedge scripting interface is relatively large involving nearly 200 functions and subroutines and roughly 3000 variables and parameters. To support the FACETS physics interface and to provide Uedge users the access to which they are accustomed, the direct Babel interface needed to provide access by name to all the 3000 variables and parameters. It took several development/test iterations to generate an efficient, buildable, moderately sized interface.

4. Other Accomplishments

The other accomplishments focus on general framework contributions, an initial parallel-load balancing exercise, and porting FACETS to LCF machines. Through discussions and teleconferences, I have contributed to the FACETS framework design, and I have written the interface specification in SIDL for components being connected using Babel.

I also lead a team exercise to evaluate the FACETS load-balancing challenges. There are different physics modules available to model core and edge physics, and each has different computational requirements and scalability properties. By considering the computation requirements and scalability of each potential algorithm, we were able to recommend models that are roughly load balanced with respect to each other. This exercise indicated that the wall model is the least computationally intensive, so it is unlikely to ever be the computational bottleneck.

Lastly, I have worked closely with team members from Tech-X Corporation to get FACETS building and running on clusters and LCF machines. This involved getting Babel to install on Franklin, Bassi, and Jacquard at NERSC. Porting to LCF machines is always a challenge because each machine does things in a slightly different way which requires modifications to the configuration and build system. In addition, we did lots of team debugging to get the statically linked Uedge building and running on clusters.